Das Märchen von der verteilten Terminierung

[F. Mattern - Informatik-Spektrum 8:6, pp. 342-343, 1985]

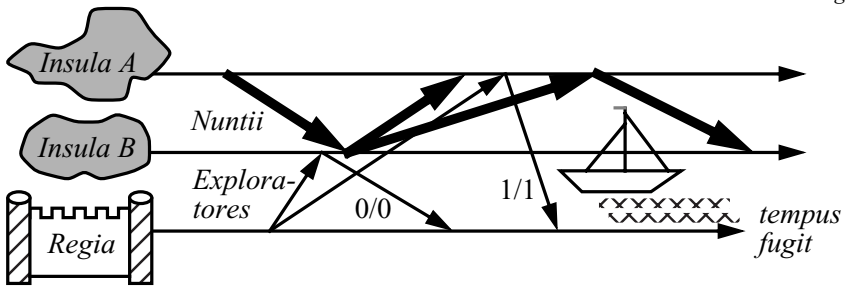
Als einst der König von Polymikronien merkte, dass die Zeit gekommen war, sein aus unzähligen Inseln bestehendes Reich gerecht unter seinen Enkelkindern aufzuteilen, sandte er Botschaften an die weit über das Land verteilt lebenden Weisen aus, auf dass diese ihm einen klugen Vorschlag unterbreiten mögen.

Wohl wusste der König um den eigenwilligen aber steten Lebenswandel seiner Ratgeber, welche den ganzen Tag nichts taten, als zu essen und zu denken: War einer von ihnen bei Speis und Trank, so konnte alleine eine königliche Botschaft oder eine Nachricht eines anderen Weisen ihn zum Denken anregen - er griff meist sogleich zu Feder, Papier, Tinte und Siegel, um einigen übrigen Mitgliedern des so weit verteilten königlichen Konsiliums eine neue Weisheit zuzusenden. Hungrig vom Denken wandt er sich alsbald wieder der stets fürstlich gedeckten Tafel zu.

Als indes die Jahre vergingen und der König immer älter wurde, ohne dass er von den Weisen einen Rat erhalten hätte, seufzte er, schickte nach seinem geheimen Hofrat und sprach zu ihm: "Ich weiss wohl, wie schwierig ein gerechter Plan zur Aufteilung meines Erbes ist, und ich kenne die Regeln meines Konsiliums, wonach man mir erst kundtut, wenn die königliche Sache so erschöpfend beraten wurde, dass ein jeder der Weisen zufrieden ist und keine Botschaft mehr unterwegs ist. Alleine die königliche Post bereitet mir Sorge - ist etwa ein Schiff in den Stürmen der Meere gesunken oder hat sich ein Bote in der Weite des Reiches verirrt?"

"O königliche Hoheit", entgegnete der geheime Hofrat und sprach weiter: "Unermesslich gross ist Euer Reich, und gar lange brauchen die Segler, um von einem Eiland zu einem anderen zu gelangen. Rein niemand vermag die Zeit vorher abzuschätzen, und es wird sogar berichtet, dass in der ein oder anderen Nacht ein Postboot ein anderes überholt. Aber die Segelkünste der Seefahrer, die hohe Schule der Schiffsbaumeister und die pflichtbewusste Ergebenheit Eurer Diener sorgen dafür, dass nicht eine einzige Botschaft verloren gehen kann. Lasset uns also Kundschafter aussenden, mein König, um jeden Ratgeber zu befragen, wieviele Botschaften er empfangen und versandt hat. So können wir leicht Gewissheit darüber erlangen, ob summa summarum so viele Nachrichten versandt wie empfangen wurden und das Konsilium des Königs Sache abschliessend beraten hat."

Fortsetzung--->



Übungen (2) zur Vorlesung "Verteilte Algorithmen"...

a) Man beweise die Korrektheit des im Märchen beschriebenen "Doppelzählverfahrens" zur Feststellung der verteilten Terminierung.

b) Man beweise die Korrektheit des Echo-Algorithmus:
- der Initiator terminiert erst, wenn alle Knoten informiert wurden ("safety"),
- nach endlicher Zeit terminiert der Initiator ("liveness").

Überlegen Sie sich, was für Beweistechniken Sie einsetzen können (Invarianten, Induktion...) und wie genau / formal die Spezifikation des Algorithmus sein sollte, damit Sie im Beweis formal argumentieren können. Geben Sie ggf. eine formalere Spezifikation des Algorithmus an.

---> Fortsetzung Märchen

Der König war hoch erfreut über diese weisen Worte, schöpfte neue Zuversicht und beauftragte sogleich den Hofmathematicus, einen Plan auszuarbeiten. Dieser erschien alsbald mit einer grossen Leinwand und sprach: "Majestät, auf diesem Szenario seht Ihr, dass des Hofrats Plan versagen kann - die zu den Eilanden A und B gesandten Kundschafter berichten, dass so viele Botschaften empfangen wie versandt wurden - summa summarum nur eine Botschaft. Nichtsdestotrotz sind noch Nachrichten unterwegs. Die Sache ist wohl so, dass die Kundschafter sehr klug vorgehen müssen, um sich nicht täuschen zu lassen und des Königs Zählung zu verfälschen, alldieweil wir primo die Uhr noch nicht erfinden konnten und somit auch keine einheitliche Reichszeit haben, secundo wir den Rundfunk noch nicht kennen und tertio die ehrwürdigen Sitten es verbieten, dass die Weisen an einem gemeinsamen Ort zusammenkommen." Der König war erstaunt über die gar wundersamen Worte und meinte: "Nun, das weiss ich wohl, denn ich bin der König. Was also rät Er mir ?" "Hoheit, mein Plan sieht vor, die Kundschafter erneut auszusenden, sobald der letzte bei Hofe eingetroffen ist. Wird alsodann das Ergebnis genau bestätigt und sind die Summen gleich, so ist keine Nachricht mehr unterwegs." "Vortrefflich", meinte der König, der nichts verstanden hatte. "Kümmere Er sich nur sogleich um die Instruktion der Kundschafter!"

Der Hofmathematicus tat wie befohlen, verbesserte seinen Plan noch verschiedentlich, und bald waren die Kundschafter mit allen königlichen Vollmachten versehen auf den besten Seglern des Reiches unterwegs zu den Weisen.

Als endlich der Schluss der Weisen bei Hofe eintraf, war der König so voll Glück, dass er den Hofmathematicus bald darauf zu seinem ersten Hofinformaticus ernannte. Und dieser forscht, wenn er nicht gestorben ist, noch heute in einem stillen Turm des königlichen Palastes... an einer noch besseren Lösung zum Problem der verteilten Terminierung!

Papier war kostbar - so verzichtete der Hofinformaticus leider darauf, einen Korrektheitsbeweis seines Planes niederzuschreiben. Einer Randbemerkung seiner Schriften entnehmen wir, dass er später ein Verfahren ersann, bei dem nicht in jedem Fall die Inseln zwei- oder mehrfach von den Kundschaftern aufgesucht werden müssen. Desweitern gelang es ihm offenbar, die Zahl der notwendigen Kundschafterfahrten durch eine einfache Funktion der Zahl der Inseln und Botschaften zu beschränken. Leider reichte wiedereinmal der Rand zur Darstellung der Methode nicht aus. Wer hilft mit bei der Rekonstruktion und Verifikation der Verfahren ?

Zeitkomplexität

Beachte: Algorithmen i.a. nichtdeterministisch --> *mehrere* mögl. Berechnungen!

Variable Zeitkomplexität eines vert. Algorithmus = max. "Zeit" aller Berechnungen des Algo unter:

Z1: Lokale Berechnungen erfolgen in Nullzeit
Z2: Eine Nachricht benötigt *maximal* 1 Zeiteinheit

Einheitszeitkomplexität eines vert. Algorithmus = max. "Zeit" aller Berechnungen des Algo unter:

E1: Lokale Berechnungen erfolgen in Nullzeit
E2: Eine Nachricht benötigt *exakt* 1 Zeiteinheit

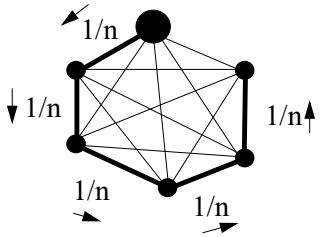
Behauptung:
Es gilt *nicht* immer variable Zeitkplx ≤ Einheitszeitkplx.

- Grund: Einheitszeitkplx erlaubt nicht alle Berechnungen!
- Frage: Gilt Umkehrung?

Bsp. Echo-Algorithmus auf vollständigem Graph

(1) Einheitszeitkomplexität = 3 ← Phase 1: Alle werden rot
Phase 2: Alle bis auf Initiator werden grün
Phase 3: Initiator wird grün

(2) Variable Zeitkomplexität ≥ n



- Explorer "aussen": 1/n Zeiteinheiten
- Jede sonstige Nachricht 1 Zeiteinheit
- Entarteter Baum Tiefe n-1 nach einer Zeiteinheit aufgebaut
- Echo beim Initiator nach n Einheiten

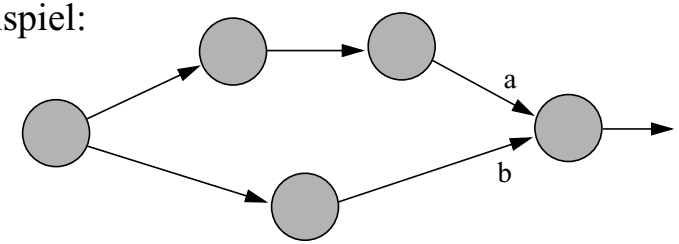
Zeitkomplexität: Welche Definition?

- *Einheitszeitkomplexität*: Einige Berechnungen bleiben unberücksichtigt!

Nicht bei var. Ztkplx! (Wieso?)

unwahrscheinliche?

Beispiel:



Trifft a vor b ein --> sehr lange Berechnung, sonst terminiert

mag vielleicht in 10% aller Fälle der Fall sein...

aber wie oft wirklich?

systemabhängig!

- *Variable Zeitkomplexität*: Resultat wird u.U. durch extrem unwahrscheinliche Berechnungen bestimmt

- Für worst-case: variable Ztkplx? Aber: average case?

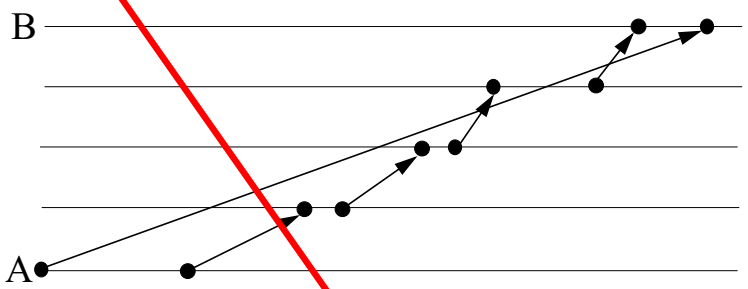
- Genauer: Wahrscheinlichkeitsverteilung --> Erwartungswert
  - systemabhängig
  - schwierig
  - jeden Tag anders...

Ein anderes Zeitkomplexitätsmass

Längste Nachrichtenkette einer Berechnung

Dann max. (oder Mittelwert?) über alle Berechnungen

Beispiel:



Sende erst direkt, dann indirekt an B

- Prozess A initiiert den Algorithmus
- Beendet, wenn B direkt oder indirekt von B hört
  (Also: Wenn B eine Nachricht empfängt)

- Einheitsztkplx. = 1
- Var. Ztkplx. = 1
  (worst case)
- Längste Kette = 4

grösser!

- Wie sinnvoll ist dieses Mass?
- Was ist das "richtige" Mass für die Zeitkomplexität?

Bem.: solche Ketten spielen im Sinne eines "critical path" bei Beschleunigungsuntersuchungen auf Parallelrechnern eine Rolle

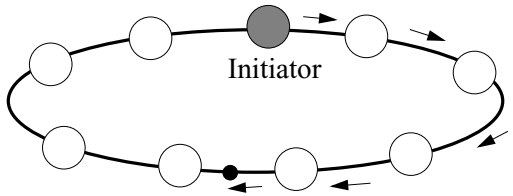
Broadcast auf speziellen Topologien

- Echo-Algorithmus realisiert einen Broadcast

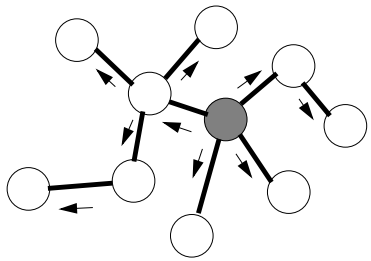
auf bel. zusammenh. Topologie

  - Verteilen von Information ausgehend von einem Initiator
  - für beliebige (zusammenhängende) Topologien
  - liefert sogar "Vollzugsmeldung" durch Echo-Nachrichten

- Auf *speziellen* Topologien lässt sich der Broadcast auch effizienter realisieren
  - Beispiel *Ring*: Ein "Token" zirkuliert mit der Information; alle sind informiert, wenn das Token wieder beim Initiator eingetroffen ist
  - ggf. kann einer anderen Topologie ein Ring überlagert werden



- Beispiel *(Spann)baum* (tatsächlich Unterschied zum Echo-Algorithmus?)



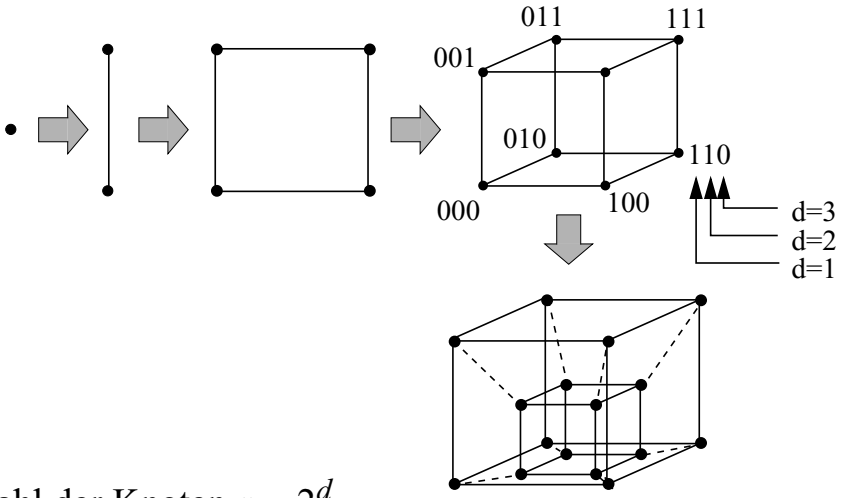
vorausgesetzt wird jeweils, dass der Algorithmus "weiss", dass eine spezifische Topologie vorliegt!

- Beispiel *vollständiger Graph* (als Denkübung)

Hypercubes

- Hypercube = "Würfel der Dimension d"
- Rekursives Konstruktionsprinzip
  - Hypercube der Dimension 0: Einzelrechner
  - Hypercube der Dimension d+1:

    „Nimm zwei Würfel der Dimension d und verbinde korrespondierende Ecken“



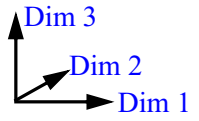
- Anzahl der Knoten $n = 2^d$
- Anzahl der Kanten = $d\ 2^{d-1}$ (Ordnung O(n log n))
  - viele Wegalternativen (Fehlertoleranz, Parallelität!)
  - maximale Weglänge: d = log n
  - mittlere Weglänge: d/2 (Beweis als Denkübung!)

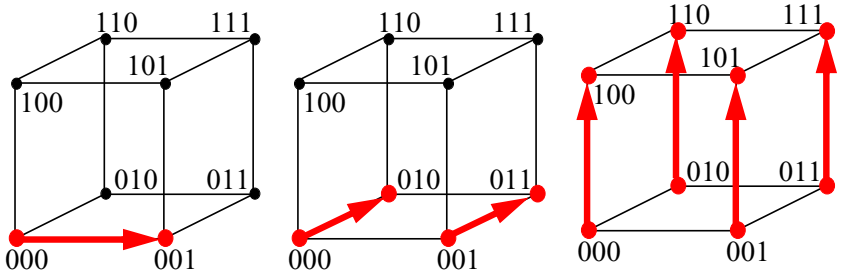
wieviele verschiedene Wege der Länge k gibt es insgesamt?

- Knotengrad = d (nicht konstant bei Skalierung!)
- Einfaches Routing von einzelnen Nachrichten
  - xor von Absende- und Zieladresse...

Broadcast in Hypercubes (1)

- Initiator habe die Nummer 00...00 (binär)
- Wir verzichten hier auf Vollzugsmeldung
  (also keine Acknowledgemnents oder Endeerkennung)





- Analog zum rekursiven Aufbau des Hypercube:
  - zunächst in Dimension 1 senden: Teil-Hypercube der Dimension 1 ist damit informiert
  - dann senden alle Knoten der Dimension 1 in Dimension 2
  - dann Dimension 3 etc.
- Nach d "Takten" sind alle Knoten informiert
  - Zeitkomplexität ist daher d (unter welchem Zeitmass?)
  - Nachrichtenkomplexität: $1 + 2 + 4 + ... + 2^{d-1} = 2^{d-1}-1 + 2^{d-1} = n-1$
    (jeder Knoten, ausgenommen der Initiator, erhält genau eine Nachricht)
- Welche Komplexität hat ein *optimaler* Broadcast-Algo.?
- Geht es besser?

was heisst überhaupt "besser"?

  - Algorithmus startet ziemlich "langsam": am Anfang geschieht wenig parallel!
  - Kann man dies durch *gleichzeitiges* Versenden "in alle Richtungen" beschleunigen?

Broadcast in Hypercubes (2)

- Ein anderes Verfahren (Vergleich als Denkübung!)

- Initiator sendet an alle seine Nachbarn:

  0...01, 0...010, 0...100, ... , 10...0

  in "kanonischer" Numerierung

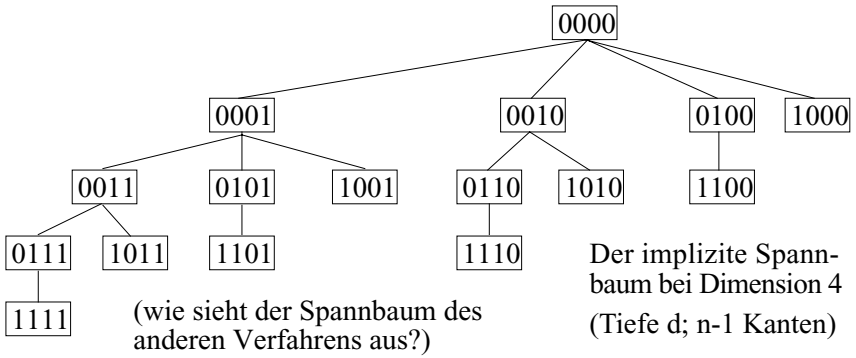
  am besten gleichzeitig, wenn dies technisch geht!

- Ein Knoten mit der Nummer 0...01x...y...z leitet die Information an alle seine "höheren" Nachbarn weiter:

  linkeste 1 | beliebiges Restmuster

  0...00**1**1x...y...z
  0...0**1**01x...y...z
  0...**1**001x...y...z
  ...
  10...001x...y...z

  Von welchem (eindeutigen) Knoten A wird Knoten B informiert?
  Setze *vorderste 1* von B auf 0
  --> = Nummer von A



(wie sieht der Spannbaum des anderen Verfahrens aus?)

Der implizite Spannbaum bei Dimension 4
(Tiefe d; n-1 Kanten)

- Der Algorithmus wird z.B. in Mehrprozessorsystemen verwendet
- Wie effizient ist der Algorithmus? (Geht es besser?)
- Denkübung: Formuliere Algorithmus für einen beliebigen Initiator (schliesslich sind Hypercubes symmetrisch...)
- Denkübung: Vergleich mit Flooding bzw. Echo-Algorithmus

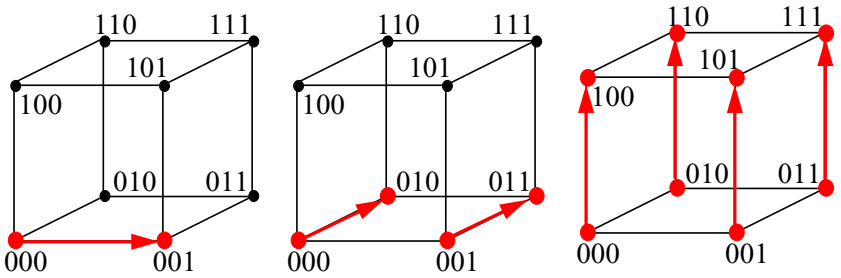
Noch ein anderer (besserer?) Algorithmus

- Beobachtungen:
  - Ein Baum verwendet im Hypercube relativ wenig Kanten --> schlechte Ausnutzung potentielle Parallelität
  - Es gibt *mehrere* Spannbäume in Hypercubes --> diese nutzen?

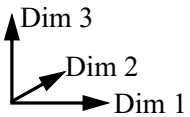
- Sender 0...0 teilt die Nachricht in d Pakete
- Sender startet für jedes Paket eine eigene "Welle":
- *1. Paket* in *Dimension 1* senden --> 0...01
- Dann: Alle informierten Knoten (also 0...0 und 0...01) senden dieses Paket in Dimension 2
- Etc. Welle für Paket 1 breitet sich analog zur rekursiven Definition des Hypercubes in einer jeweils zusätzlichen Dimension aus
- Das *2. Paket* wird erst in *Dimension 2*, dann 3,..., d und erst zuletzt in Dimension 1 gesendet
- Das *3. Paket*: Dimensionsreihenfolge 3, 4, ..., d,1, 2
- Etc.: das *d.-Paket* in Dimensionsreihenfolge d, 1, 2,..., d-1

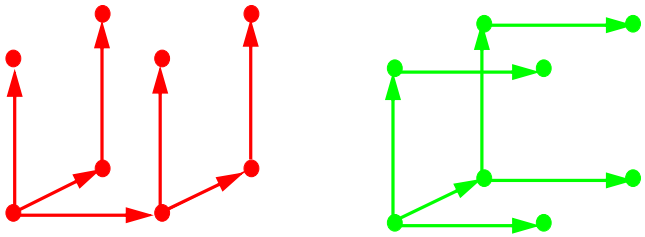
Denkübungen:
- Können so die Pakete gleichzeitig verschickt werden?
- Ist dann in jedem "Takt" pro Kante nur eine Nachricht unterwegs?
- Wieviele (kantendisjunkte ?) Spannbäume gibt es in einem Hypercube?

Veranschaulichung des Algorithmus



Die drei “Takte” der Welle von Paket 1

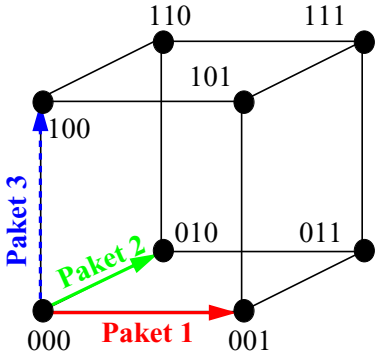




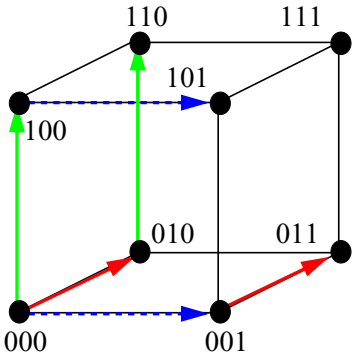
Die Spannbäume bzgl. Paket 1 und Paket 2

Parallelausführung der drei Wellen

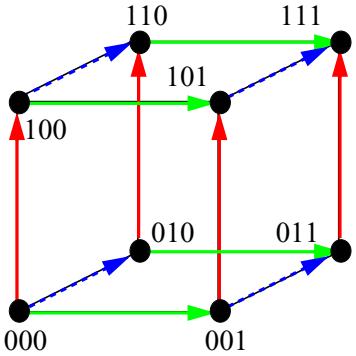
Takt 1

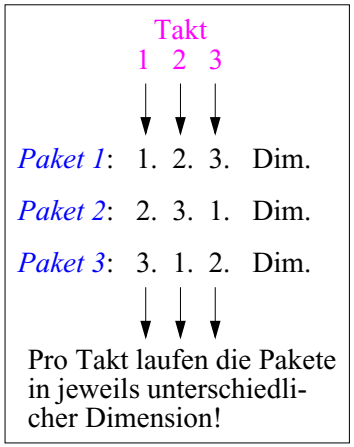


Takt 2



Takt 3





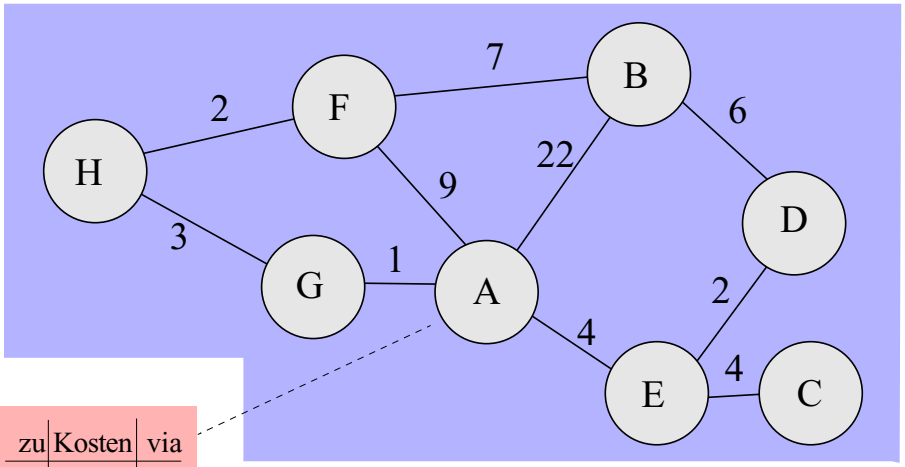
Es können also tatsächlich die drei Wellen parallel ausgeführt werden, ohne dass diese sich gegenseitig stören!

--> Dies ist das (im Prinzip) schnellere Verfahren!

Beachte: Ein globaler Takt ist gar nicht nötig!

Verteilte Berechnung von Routingtabellen für kürzeste Wege

Gegeben: ungerichteter zusammenhängender Graph mit bewerteten Kanten (Kosten, Länge...)



zu	Kosten	via
A	0	-
B	22	B
C	∞	?
D	∞	?
E	4	E
F	9	F
G	1	G
H	∞	?

Anfangs-tabelle für Knoten A

- Jeder kennt anfangs die Kosten zu seinen Nachbarn
- "Spontanstart": Sende eigene Tabelle an Nachbarn
- Bei Empfang einer Tabelle über Verbindung mit Kosten g:
  Für alle Zeilen i der Tabelle:
  Falls Nachricht.Kosten[i]+g < Knoten.Kosten[i]:
  ersetze Zeile (Kosten := Kosten+g; via := Absender)
- Falls sich Tabelle verändert hat:
  Neue Tabelle an alle Nachbarn (Ausnahme: Sender)
- Wie Terminierung feststellen?

- Ist eine verteilte Version des Bellman-Ford-Algorithmus
  - ähnlich dem bekannten Dijkstra-Algorithmus für kürzeste Wege
  - " Relexationsprinzip" (Bellman 1958, Dijkstra 1959, Ford 1962)

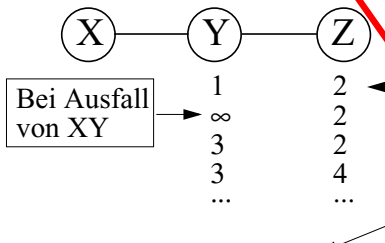
Kürzeste Wege in Rechnernetzen

- Algorithmus wird oft als dynamisches ("adaptives") Routing-Verfahren verwendet, wo in regelmässigen Zeitabständen die Tabellen neu berechnet und ausgetauscht werden
  - bzw. dann, wenn sich etwas ändert (Kosten einer Verbindung, z.B. Ausfall einer Leitung oder Änderung der Lastsituation)

- Metrik für die Kosten z.B.:
  - (gewichtete)Anzahl der hops
  - Bitrate einer Verbindung
  - Verzögerung einer Verbindung (z.B. gemessen mit Testpaketen)
  - Länge der Warteschlange vor einer Verbindung

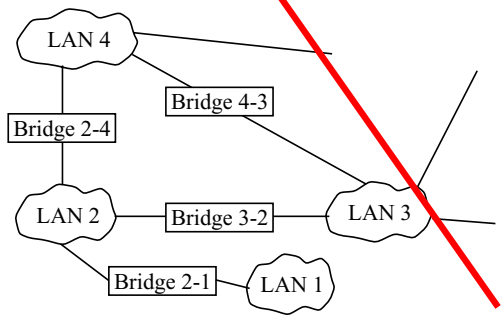
- "Count to infinity-Problem"

mehr zu diesen Dingen in anderen Vorlesungen ("Rechnernetze")



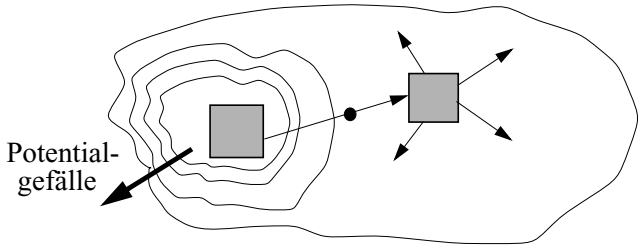
- Durch die kürzesten Wege zu einem festen Knoten ("Wurzel") ist ein kostenminimaler Baum gegeben

- Algorithmus wird in LANs eingesetzt, um einen Spannbaum zu bestimmen (Knoten = Teil-LAN; Kante = Bridge)
- Zyklenfreiheit ist wichtig, da kein Routing in LANs
- Ende wird heuristisch durch Abwarten einer Zeitspanne festgestellt



Das Paradigma der vert. Approximation

Prinzip:
- Anfang: Informiere alle Nachbarn spontan
- Bei Empfang einer Nachricht:
  - berechne neue Approximation
  - falls diese "besser": informiere Nachbarn



- *Nachrichtengesteuert* (aber "Spontanstart")
- Alle Prozesse arbeiten gleich, alle sind beteiligt
- *Nichtdeterministischer* Ablauf, determin. Ergebnis
- Beliebige stark zusammenhängende Topologie
- Assoziative Operatoren (min, max, $\cap$, $\cup$, +, and, or, ...)
- *Stagnation* bei globalem Gleichgewicht ("Optimum")
  --> Potentialunterschiede ausgeglichen
  --> Terminierungsproblem

Beispiele ("Instanzen der Algorithmenklasse"):
- ggT
- Zahlenrätsel
- Verteilen von Information ("Wissensausgleich")
- Routingmatrizen (inkl. Spannbaum)
- Maximale Identität ("election") } (noch) nicht behandelt
- Lastausgleich (Approx. eines dyn. Optimums) } (noch) nicht behandelt
- Relaxationsverfahren (Lösen von DGL) } (noch) nicht behandelt